Method of Optimization of CPU and Chipset Performance by Support of Optional Reads by CPU and Chipset

BACKGROUND OF THE INVENTION

[0001] Computer systems are composed of one or more central processing units (CPUs), a bus or other interconnect fabric, memory controllers, and input/output (I/O) adapters. These resources are subject to contention; and management of that contention is necessary to optimize performance. Specifically, "latency," the delay between the initiation of an operation and its completion, and "bandwidth," the aggregate volume of operations, are frequently traded against one another. CPU designs have thus attempted to reduce apparent latency by performing "speculative" operations, that is, operations initiated early before all information is present as to the necessity or specific parameters of the operation. Prefetching and branch prediction are examples of this speculation. Such speculation may either be explicit, under control of program instructions, or implicit, via automatic operation of the CPU logic. These speculative operations increase demand on available bandwidth. If the excess demand is too great, unbalanced system operation due to resource contention ("hotspots") could result, perhaps to the point of reducing overall system performance to below the point that would have been obtained had speculation not occurred. Thus, there is the need for system-wide management of speculative transactions.

[0002] One problem with prefetching in the prior art is that much of the data read through speculation is not actually used. That is, a speculative read may issue even though the requester is not assured that the data will be executed or the subject of a load. Prior art CPUs and chip-sets nevertheless treat speculative memory traffic and non-speculative memory traffic the same, even though speculative data may cause congestion and memory latency for non-speculative requests. This problem is exacerbated with high bus utilization; for example, it has been shown that speculative traffic may account for 50% or more of bus traffic.

[0003] It is, accordingly, one object of the invention to provide methods and systems for processing speculative requests in a way to reduce congestion and collisions with non-speculative traffic. Other objects of the invention are apparent within the description that follows.

SUMMARY OF THE INVENTION

[0004] The following U.S. Patents provide useful background to the invention and are accordingly incorporated herein by reference: U.S. Patent No. 6,292,871; U.S. Patent No. 5,778,219; U.S. Patent No. 5,761,490; U.S. Patent No. 5,721,865; U.S. Patent No. 5,566,324; U.S. Patent No. 4,980,823.

[0005] In one aspect, the invention provides a method of identifying speculative memory requests to reduce congestion or saturation in bus traffic or within a CPU or chipset. Such instructions are tagged with identifying bits (denoted herein as the "speculative ID") readable by logic utilizing the instructions in order to distinguish the request as speculative. By way of example, a CPU or chipset processing the request are programmed to identify the speculative ID. As needed, that CPU or chipset may dispose of or ignore the speculative request in preference of minimizing bus traffic and/or to preferentially process non-speculative reads.

[0006] The invention of another aspect identifies speculative requests in memory request transaction protocols. Transactions that are speculative are considered "optional" by the CPU, or chipset, and may be terminated without returning data. Speculative read requests are then preferentially ignored, or in some cases destroyed, should the CPU bus become saturated or congested. In one example of the invention, a transaction may be aborted if the request receives an error, rather than trying to recover or retry the request.

[0007] In another aspect, the invention provides a CPU or chipset architecture that processes transactions and identifies whether the transaction is speculative or not. The architecture preferably is of the type utilizing prefetch logic that decodes the transaction in order to distinguish memory system accesses that are speculative (e.g., prefetched) as compared to non-speculative "demand" memory accesses required for forward progress. When the architecture detects congestion or saturation, either in internal processing or within bus traffic, or within target memory, the speculative transaction is ignored or aborted to facilitate other more important transactions.

[0008] In yet another aspect, the invention provides a method for monitoring bus requests to identify speculative reads versus mandatory reads, in order to enhance transaction throughput. A bus controller, switch or other logic device coupled with the bus elects either to carry out speculative transactions, or not, depending upon traffic or saturation conditions on the bus or within target memory. In one example, therefore, a speculative load request to read data out of memory and to load data into registers within the register file may be ignored,

as needed. In accord with the invention, a CPU on the bus is programmed to function properly when a speculative request is not processed. Aborting a transaction preferably also includes the step of returning a response in appropriate bus protocols; accordingly, such a response may be used to notify devices desiring reply from the speculative request so as to proceed to other processing.

[0009] As used herein, a "request" can for example be an instruction, a load (including explicit or implicit loads), a message, or an operational request. As used herein, an "interconnect" can for example be a bus, a crossbar, a switch, and a point-to-point protocol.

[0010] In accord with another aspect, the invention provides a method of asserting a priority designation to speculative requests. Logic devices decoding the requests determine the priority and order processing based on the priority. A CPU connected to the logic devices may alter the priority of certain requests to improve performance.

[0011] The invention provides certain advantages. Processing architectures may be enhanced to identify and manage speculative reads so as to improve performance when bus traffic or target memory is saturated. Logic components linked to the processing architecture – e.g., switches or bus controllers, may independently decide to act, or not, on speculative transactions. A CPU or chipset device that already includes prefetch or branch capabilities may be modified or programmed, in accord with the invention, to detect and manage speculative bus requests. Such a device may for example decide that its instruction queue is too long and simply ignore the speculative request.

[0012] In another aspect of the invention, a method is provided for processing a request, such as an instruction, message or operational request, through a processor. The method includes the step of determining whether the request is speculative or not based upon a first identifier. The first identifier may for example reside as one or more bit fields within an instruction or transaction. The method further includes the steps of assessing one or both of interconnect and target resource conditions in the event that the request is speculative, and either processing the request, or not, as a function of the conditions. Assessing interconnect resource conditions may for example include assessing bus congestion and/or utilization, crossbar congestion and/or utilization, and/or point-to-point link utilization. Assessing target resource conditions may for example include assessing memory utilization and/or buffer space utilization.

[0013] In another aspect, the method may further include the step of decoding the first identifier as a speculative ID within the request. Another step of the method can include the step of encoding a bit field within the request to define the request as speculative or not.

[0014] In yet another aspect, the method may include the step of determining a priority of the request based upon a second identifier, in the event that the request is speculative. As such, the step of processing the request includes processing the request, or not, based upon the conditions and the priority. The priority identifier may be determined, for example, by decoding the second identifier as a second bit field within the request. Another step of the method can include the step of encoding the second bit field within the request to associate the request with a priority.

[0015] A typical request of the invention is a memory read request or a memory load request. The step of determining whether one of these requests is speculative may be accomplished by a CPU, chipset, bus controller, memory controller, or other logic device, to determine whether the request is speculative. In a particular advantage of the invention, this CPU, chipset, bus controller and memory controller may function independently to control the step of processing the request based on the conditions. That is, these devices do not necessarily require separate instruction so as to ignore, process or abort a speculative request.

[0016] In preferred aspects of the invention, the step of assessing target resource conditions includes assessing one or more of memory utilization, memory congestion, buffer space utilization, and bus congestion.

[0017] In preferred aspects of the invention, the step of assessing interconnect conditions includes assessing one or more of bus utilization, bus congestion, crossbar utilization, cross bar congestion, and point-to-point link utilization.

[0018] The invention also provides an improvement to CPU architecture of the type that initiates both speculative and non-speculative memory requests. This improvement includes decode logic, to determine whether the requests are speculative, and assessment logic, to determine one or both of interconnect and target resource conditions. The improved CPU architecture then processes speculative requests, or not, as a function of the conditions.

[0019] In one aspect of the invention, the improved CPU architecture includes a prefetch unit that prefetches speculative requests; the decode logic then detects whether prefetched requests are speculative.

[0020] The invention further provides a system for processing speculative requests. One or more requests in the system have bit fields defining the requests as speculative or non-

speculative. Decode logic within the system decodes the bit fields to determine whether one or more requests are speculative. And processing logic within the system processes speculative requests, or not, based on at least one of interconnect and target resource conditions. By way of example, one or both of the decode logic and processing logic may include a CPU, a chipset, a bus controller and/or a memory controller.

[0021] In a further aspect, the system includes a bus controller for assessing one or more of bus congestion and bus utilization conditions.

[0022] The invention is next described further in connection with preferred embodiments, and it will become apparent that various additions, subtractions, and modifications can be made by those skilled in the art without departing from the scope of the invention.

BRIEF DESCRIPTION OF THE EMBODIMENTS

[0023] A more complete understanding of the invention may be obtained by reference to the drawings, in which:

[0024] FIG. 1 shows a system processing architecture constructed according to the invention;

[0025] FIG. 2. illustrates one instruction format to identify speculative instructions with optional priority, in accord with the invention; and

[0026] FIG. 3 shows a flowchart for processing speculative requests in accord with the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0027] FIG. 1 illustrates a processing architecture 10 of the invention. A CPU 12 connects to a memory 14 by a system bus 16 that carries data and control signals between CPU 12 and memory 14. Memory 14 may for example store instructions to be performed by CPU 12. Instruction cache 18 stores these instructions, and other CPU-generated instructions, for processing within CPU 12. FIG. 2 illustrates one speculative instruction format 100 suitable for use with the invention.

[0028] CPU 12 has one or more register files 22. Generally, register files 22 include an integer register file 22a to store integer values, a floating point register file 22b to store floating point values, and a speculative register file 22c to store speculative or temporary data. CPU 12 processes instructions through an operational unit 24, often in the form of multiple instruction pipelines 24(1)-24(n). Each of instruction pipelines 24(1)-24(n) may invoke

branch predictions and speculative executions resulting in speculative data within registers 22c. A speculative load may for example be explicit or implicit, and loaded ahead of a load that is dependent upon execution of a conditional branch. This data is "speculative" since it may not be permanently used by a subsequent load instruction. If the data is actually used, the subsequent load architects the memory and that data is no longer speculative. This subsequent load execution may acquire its data from cache, memory or a register file.

[0029] To reduce latency within CPU 12, a prefetch unit 26 fetches certain instructions based on speculation that data from those instructions may be used; such data may be stored within speculative registers 22c. A bus controller 28 monitors traffic on CPU bus 30 and between various devices 22, 24, 26.

[0030] FIG. 2 illustrates a format for one 64-bit speculative instruction 100 utilized in accord with the invention. Instruction 100 has 64 bits segmented into n bits of the instruction field 102, providing instruction opcode and associated operands, and m bits of the precode control field 104, indicating whether instruction 100 is speculative or not (i.e., the speculative ID). By way of example, the n bits of instruction field 102 may define operand address information, register file addressing information, conditional information, control bits and branch value; precode field 104 may indicate that instruction 100 is a speculative memory read. Optionally, precode field 104 also has reserve bits 106 used to specify a priority for instruction 100.

[0031] To utilize instruction 100, CPU 12 identifies instruction 100 as a speculative request by decoding precode field 104. In the event instruction 100 is not a speculative request, processing of instruction 100 continues within CPU 12 per normal. In the event instruction 100 is speculative, the instruction may be ignored or aborted, effectively delaying, sometimes permanently, the architecting of data associated with the instruction. In one example, bus controller 26 decodes instruction 100 as a speculative read request and determines that traffic on bus 30 is too congested to process instruction 100, delaying processing until a "retry" at a later time. In another example, CPU 12 determines that target memory within register file 22 is saturated and ignores a speculative load instruction 100.

[0032] FIG. 3 illustrates a flowchart 200 for processing an instruction 100 through a processing architecture of the invention, e.g., through architecture 10. An instruction transaction starts at step 202. By way of example, the transaction may be a memory read request. At step 204, the request is received, such as by a pipeline within operational unit 24. Logic decodes the instruction, at step 206, to decode the speculative ID to determine whether

the request is speculative. By way of example, the logic may be a CPU, chipset, bus controller, switch or other logic handling the request. If the request is not speculative, the request is processed at step 208. If the request is speculative, the logic optionally decodes reserve bits 106 to determine a priority for the request, at step 210. The logic then assesses bus traffic, or target memory, at step 212, to determine conditions associated with the request. At step 214, the logic determines whether the request should be processed in view of those conditions. If the logic decides that conditions do not warrant processing of the request, such as during a memory saturation condition, the logic decides whether to retry the request later at step 216. If yes, the request is again processed at step 212; if no, the request is aborted at step 218. In the event that step 214 determines that conditions support continued processing of the speculative request, the request is processed at step 208. By way of example, a speculative memory request may process, after step 214, to store data within temporary memory registers 22c, at step 208.

[0033] Speculative transactions are thus "optional" as determined by the logic. As such, the request may be terminated without returning or storing data. Speculative read requests may be ignored or retried, and in some cases aborted depending on bus or target conditions. A request may be similarly aborted due to an error indication, at step 218. A priority code in reserve bits 106 may change the decision at 214 relative to current bus or target memory conditions. CPU 12 may alter the priority code of selected speculative transactions to change processing characteristics.

[0034] The invention thus attains the objects set forth above, among those apparent from the preceding description. Since certain changes may be made in the above methods and systems without departing from the scope of the invention, it is intended that all matter contained in the above description or shown in the accompanying drawing be interpreted as illustrative and not in a limiting sense. It is also to be understood that the following claims are to cover all generic and specific features of the invention described herein, and all statements of the scope of the invention which, as a matter of language, might be said to fall there between.